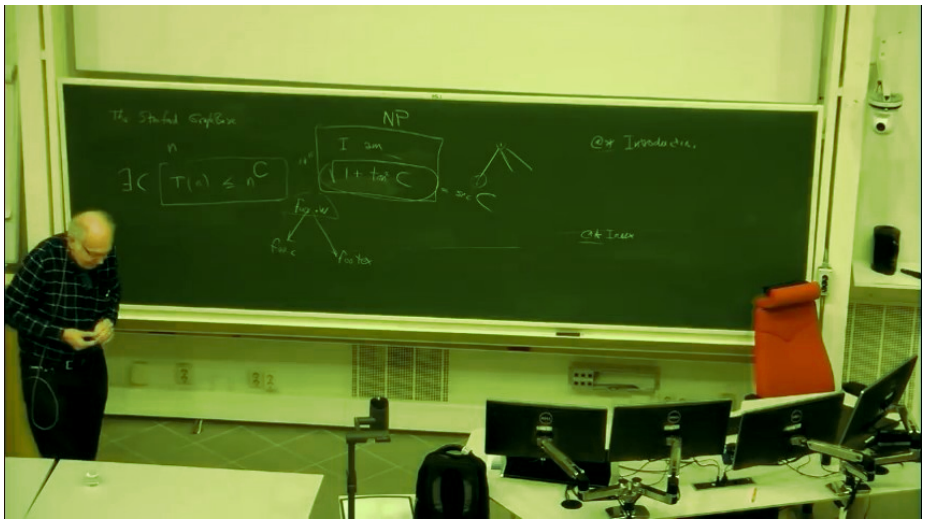
Donald Knuth at 80 : Lulea answers.

Camille Akmut

Abstract

Transcript of the 'questions and answers' of Donald Knuth at Lulea in 2018; covering topics from the pre- or early history of computer science to the fundamental problems and latest advances in theoretical computer science.



Introduction

When Donald Knuth appeared in Lulea[1] he had just turned 80^2.

He reflected on a life dedicated to science : not in the form of a speech, but inside the circumstances of answers given to questions by students and others.

"*I wanted to make you happy*", he said simply.

When he mishears some of them, unintended but always interesting effects come out of this : generative misunderstandings, results from over half a century of computer science meditations.

The present text, our transcript of this event, is given with accompanying notes and where we judged fit supplementary material.

This publication, neither an endorsement nor celebration, is presented as a document of some historical importance, if not sociological significance.

—

When Donald Knuth gives his definition of what constitutes a science, of what a science is, we know this to be the construct of an original. This does not keep us from simultaneously disagreeing.

The increasing formalism or mathematization of various disciplines has made it so that large parts of modern economics could be entered in a computer for instance.

This makes it neither a sound science, nor a science at all. Frail foundations and gross approximations make it much closer to an edifice (as Cavailles liked to say) of questionable standing and uncertain longevity.

In fact, according to some economists – of strongest mathematical abilities or backgrounds – "*economists have never made a single correct prediction*"...

—

We continue to prefer the economy of Marx and that of historical economists, over the formulae-heavy but useless of Friedman and co. (computable only to be shown wrong).

—

And, though we keep wondering about the computer scientists of tomorrow, our descriptions of them are complete. We carry so many hopes for them... They will know who came before them.

When I started teaching I was at Caltech, and one of the great professors at Caltech was Richard Feynman, the physicist, you know Nobel Prize physicist and so on. And, at the end of every class he had a tradition of questions and answers.[3]

And, so I did the same then for my classes : on the final day of class students wouldn't have to come if they didn't want to, but if they wanted to they could then ask me any questions.

Well, the rule was actually they could ask me any question except religion, politics or the final exam.

But, today if you want to ask me a question about religion or politics it's OK, but – although I might not be very good at answering questions of that kind.

But, the idea is I don't, I didn't want to come with my agenda, but I wanted to make you happy.

So, each of you who has something that you wonder of what I might think about a certain thing, this is, you know please, please ask the question.

I will try to answer it without excessive technical jargon, I'll also try to answer it quickly.

I see that there's about a 100 people here, and we got I don't know 60 minutes, so if I take 1 minute at every question that's a 100 minutes.

So, I guess I'd like to say : nobody should ask 2 questions, unless we really run out. If you really have 2 questions that you, you know, that you need answers to, whisper, whisper one of them to somebody else and let them ask it.

So, let's get started then.

Maybe you're more comfortable asking a question in Swedish than in English, in that case Ulaf will translate it for me.

(...)

So, raise your hand if you have a question, and then I will choose (...)

And, don't worry about being the first person.

Well, if there are no questions [pretends to leave, to some laughters]

(...)

— Q. (...) interest in both computer science and music. I assume at one time in life you had to choose between these two. When did that happen, and why did you choose? Because you're very talented in music as well.

Yes, well, OK so the question : I never had to choose really between computer science and music because computer science wasn't invented yet when I was, when I would had to make [a choice]. But, I had to choose between physics and music.[4]

That was 195-... Well, my last year of high-school, 1956, I lived in Milwaukee, Wisconsin[5] and I don't know if, how many computers there were in Milwaukee, Wisconsin at the time, probably 1 or 2, at the most.

But, I had a very good physics teacher, and I also had good music teachers. And, I would have gone to one of 2 colleges : one university I would have been a music major, and the other one physics.

And, well I visited them both, and the people at the music one emphasized how easy it would be if I'd go there. You know, they said "*well, if you have any problems with the exams, we can...*" You know, they don't grade very strict, that's all.

The other place, the physics one said, you know "*1 out of every 3 students fails*" – at this one. And, so I chose that one.

— Q. The concept of life-long learning. I'm trying to learn still, even in this age. Some things are easy to understand, and some are hard. And, I've been grappling with gravity, trying to understand gravity. So, my question is : Is it possible to explain Einstein's understanding of gravity to a layman (such as myself)?

Well, if it is, someone could explain it to me. (...)

In other words, I started in physics, but I basically switched into mathematics very soon. Because, I found out that, although I could get A's in my physics lessons, I couldn't understand why they asked the questions that they did.

But, mathematics classes it turned out that, that was more close to what my brain was good at. So, I changed.

And, then I discovered computers.

So, I do believe that people have different ways of organizing knowledge in their brains.

(...)

And, I discovered that I was a computer-type of a geek instead of somebody with physics.

And, even in mathematics I identified with algebra but not with geometry.

I tried to learn these other things, but some of them remained difficult for me. It's not a lack of motivation, but I think it's because of the way my head works.

So, let's say that there's 1 person in [50...]. What's the population of Lulea? 100,000? (...) So if it was a 100,000 there would be 2,000 who would be geeks like me. (...)

And, I've tried to write books that those people would really love.

— Q. What do you think makes a good teacher in computer science and computer programming?

The difference between computer science and computer programming :

Of course a rose by any other name is still a rose.

When Edsger Dijkstra gave his Turing Award lecture I think he called it something like “The humble programmer”.

But, he described himself as a programmer, on the other hand there’s no question that he was an absolute great scientist.

And, at the time he gave the lecture the word ‘programmer’ had low esteem.

It was somebody that you could always hire, but didn’t have any creativity or, you know, or elegance to their job.

—

So, when you ask about the difference between words it somehow also relates to the status of those words in peoples’ minds.

On the other hand if we look at what the words really mean...

I gave, I tried to say “What’s the difference between science and art?”, for example.

So, I looked at the history of the words, and the way people have used it.

And, I found that art is in German “Kunst”. (...) So, “kunstlich” : this means something that’s not in nature.

I mean it can as, it can be beautiful, aesthetic : the difference is whether something was created by people – which is ”Kunst” – or something that was created by God or, you know, in evolution – and that’s the opposite of art in that way.

“Artificial” for example in English.

—

Now. Then there’s the word “program”.

OK, so, let’s see... Uh : A ”program” is the embodiment, the realization of an abstract idea called “algorithm”.

An algorithm is something that is a concept like the number 2 or something like that; while if you write the number 2 down on a piece of paper then it becomes data.

So, you have an abstract thing like information, and then you represent it somehow it becomes data.

An abstract thing like an algorithm you represent somehow it becomes a program.

So, a programmer is somebody who takes an idea of some computational process and makes a concrete representation of that process.

—

Science is the other word I didn’t define.

So, science is something that we understand.

And, I like to say : when you turn an art into a science, it’s when you understand it well enough to explain it to a computer.

When I can make an algorithm that explains something.

But, if I can’t explain it to a computer, that’s where the human being is still needed, and that’s art.

So, I wrote my book called “The Art of Computer Programming”.

And, even though science advances every year (we learn more and more about how to write computer programs, for example), art also advances and gets ahead of it.

And, so people can still go beyond what we know how to explain to computers.

— Q. I wonder how hard do you think 'graph isomorphism'[67] is as a problem?

Well, Laszlo Babai has just proved that it's not in/an NP.[8]

That is : that it's definitely simpler... I mean, it's not NP-complete.

(...)

But, I'm not sure exactly how much more he proved. I think he used advanced group theory to get an upper-bound on how hard the graph isomorphism problem.

The graph isomorphism problem is :

By the way, the idea of a graph is that somebody gives you a bunch of points, and then certain pairs of points are connected, other pairs are not connected.

And, I can draw two, somebody gives me a graph over here that has a 100 points and some connections between them, another person gives me another graph, another 100 points with connections between them, and says :

"Is there a way to re-number the vertices so that actually these two are exactly the same graphs?"

And, most of the graphs that I've ever seen in my life it was easy to answer this question.

But, there are some graphs that, it's very hard to decide whether they're different, because there's so much symmetry involved that you can't (...)

And, so every once in a while you get to a, some graphs that are very hard to decide, "Are they the same or are they different?"

So, people worried. It was one of the main open problems in computer science for a long time.

As to whether that was really a hard problem or not.

And, Babai showed that it actually isn't as hard as the hardest ones.

That was a year or two ago. Professor at Chicago.

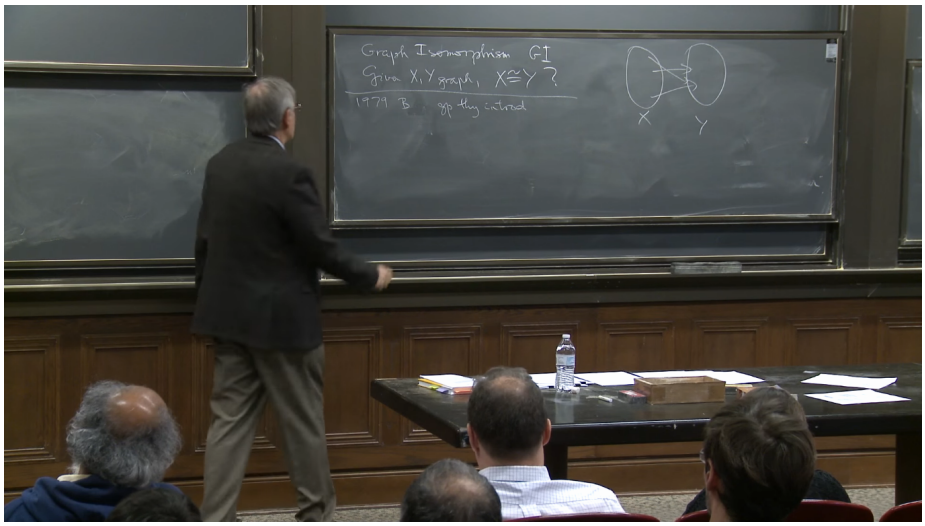


Figure 1: The problem : Given graphs X and Y, is $X \cong Y$? (Babai 2015)

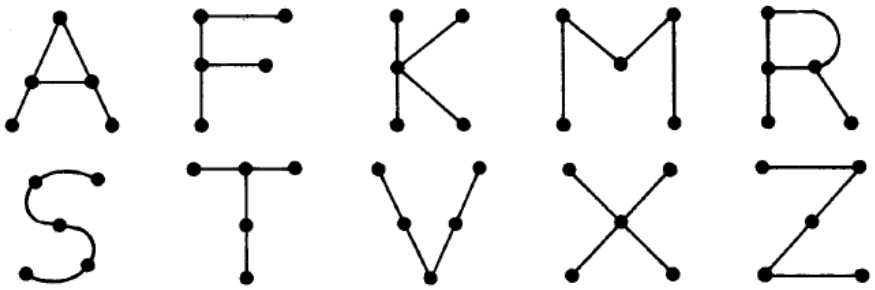


Figure 2: e.g. $A \cong R$, $F \cong T$, $K \cong X$... (Lipschutz and Lipson 2009)

— **Q. The current situation with the P = NP question.**

So, this is the the million dollar problem of theoretical computer science.

So, P is the class of problems that can be solved, for which we can write a program that solves the problem in polynomial time.

Polynomial time means that if the length of the problem is n, the time to solve the problem, the problem of size n is less than or equal to n-to-the-constant for some constant.

```
T(n) <= n^C
```

There exists a C so that the time to solve the problem is less than... :

$\exists\, C \;\; T(n) \leq n^C$

The set of all problems for which this is true is called P.

—

And, OK : then there's another class of problems called NP.

Non-deterministic polynomial time.

(...)

For example, if I ask a problem about whether a graph has a hamiltonian path.

A hamiltonian path, in a graph, is a path that goes through all points of the graph, exactly once. And, each time you go from a point to another point that's connected to it.

Nobody knows whether that problem is NP.

As far as we know, you have a big graph, there is no constant that we can always say – I mean, for every... we haven't got any algorithm that will say you're always within n-to-the-one-million-th steps.

It will say : yes there is a hamiltonian path, or not.

But, it's easy to just try all possible routes, routes through the graph (...) And, so if there is a hamiltonian path, actually by the time you've gone through all vertices you'll know, and you'll say 'Yes, there is a hamiltonian path!'. That's NP.

Now, there are other ways to describe NP;

But, the intuition is a lot of the problems that we solve all the time are based on this branching structure : try something, but we don't know which way to go.

And, if all of these problems [in NP] could be solved in polynomial time [i.e. in P], that would be pretty cool, because we could do a lot more than we know now.

—

So people set this up as a problem, and they started to say "Well, if a problem is P, it's easy."

Unfortunately, they were mistaken.

Because, for two reasons :

One is if this constant, if C is large, it doesn't help you at all.

$$\ldots \;\; T(n) \leq n^{\mathbf{C}}$$

Because, if C is uh, let's say a 100, then you can't solve the problem when n is 3.

3-to-the-100 is more than, you know, the length, microseconds in the age of the universe, or something like this.

So, just having a constant doesn't mean that it's good.

When they [people] say "polynomial time" they really mean a small constant, or exponent.

—

The question though is whether or not all of these [(problems in) P and NP] are really the same. (...)

Are these problems really different from these problems.

They certainly seem to be as far as we know.

And, there, we get to another very interesting point : the difference between existence and reali-, realization, existence and embodiment.

It might be that somebody could prove –

I tend to suspect that within the next hundred years somebody will prove that for all, all of these problems in NP can also be solved, they also are polynomial time-satisfiable in the sense that there does exist some constant, for each problem in NP.

However, I think that person will also prove that we don't know what the constant is, we only know that it exists.

And, so we won't know the algorithm, we won't know the method, we'll just know that if we were infinitely wise, and we had infinite amount of time, then we would eventually find the method.

So, just knowing that it exists means that it's in P.

But, knowing what the algorithm is, is what we can use, in life.

Everything I know is consistent with this idea.

—

There's already a problem like this where we do know a problem that is in P, but we don't know what the algorithm is.

This comes from a famous... Robertson and Seymour have a theory about graph theory, so that [for] any minor-closed family of graphs, there is an algorithm to test whether a given graph belongs to that family.

It's a technical term, but a "minor-closed family of graphs" for example includes the idea of planar graphs, a graph that you can draw on a blackboard without crossing lines.

I guess I should say what it means to be "minor", "minor-closed family" :

You define a class of graphs where you say that – if I erase any point, and everything connected to that point, then it's also a graph in the family – or, if I take two points and I collapse them together into one point, that's also a graph in the family.

And, this is true of the planar graphs : if I erase any point of a planar graph, it's still planar; if I collapse any two points together, I mean any two adjacent points together, it says planar.

But, minor-closed graphs there's a huge number of these.

And, almost none of them do we know an algorithm to test whether or not it belongs to that family of graphs.

On the other hand Robertson and Seymour showed it's always polynomial.

But, their proof does not tell us what the polynomial is.

And, it might be, this number C might be $10^{10^{10^{10^{\cdots}}}}$ (...)

Finite numbers are, can get so big that they are incomprehensible.

And, therefore knowing that there's a polynomial time algorithm tells us really nothing, to be practical; but, mathematically it's very interesting : it means that we can't prove that there isn't...

OK.

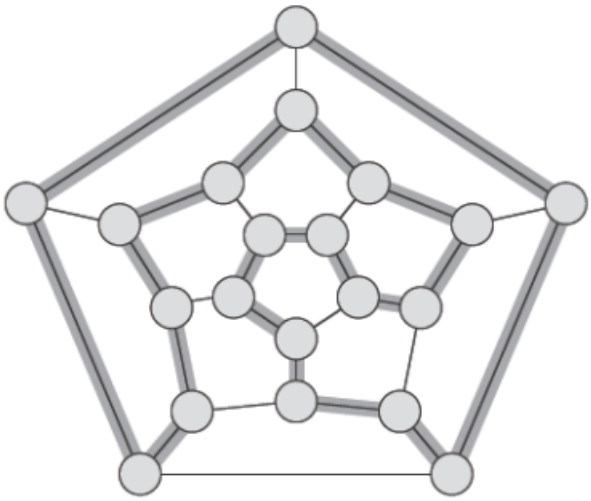


Figure 3: a hamiltonian path. (CLRS 2009)

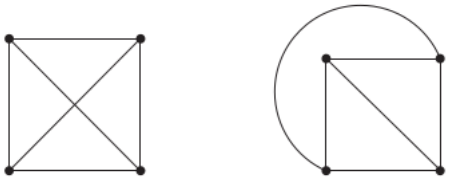


Figure 4: a graph (left) and its planar version (right). (Rosen 2011)

— Q. How do you go ahead when you try to solve a problem. What's your way of thinking?

Well, uh... I try pencil and paper.

I try small cases.

And, I almost always write a computer program in order to experiment with it.

So, I try to learn the territory.

I try to consider special cases of the problem that I know how to solve. And, then I try to go a little further.

I try to... If the problem depends on a parameter n, I try... Some people look first at n equals a million, but I tend to look first at n equals 1 or 2, maybe n equals 0.

But, I want to get firmly in mind what it is.

And, then I try to see : well, what if I play around with this problem, what if I change...

What if I have a solution to a smaller problem, can I put those solutions together to get a solution to the bigger one.

But, I'm always trying to get data so I know some things that are true about the particular thing I'm working, some things that are false that I'm working on. (...)

Learning to ask questions that get you involved with the problem. (...)

And, I tend to write 5 programs a week. Some of these programs are trivial, but other ones sometimes, you know, it might be a 20-page program, something like that. (...)

— **Q.** `CWEB`

(...) `CWEB` is a pre-processor that combines C and TeX.

So, we have a .w file, let's call it foo.w, and then out comes foo.c, which will run. And, then I have another processor to take this into foo.tex, and this is a file that I can make a listing from.

(...)

— **Q. What makes a good teacher?**

What makes a good t-shirt?

Of computer programs...

Hey. That's certainly a great question.

At my birthday party my son was wearing a t-shirt that was for mathematicians (...)

— **No, a teacher.**

Oh, a teacher! But, I liked the other question...

Figure 5: “Knuth is my homeboy” (Geekz shop)

REFERENCES

—. 2018 [2017]. "Social conditions of outstanding contributions to computer science : a prosopography of Turing Award laureates (1966-2016)"

—. 2019. ""What is computer science?" The original debates surrounding the birth of computer science and the myths born out of them (1960-1975) : a selected bibliography."

Babai, Laszlo. 2015. ""Graph Isomorphism in Quasipolynomial Time I" Seminar Lecture by Laszlo Babai on November 10, 2015". /watch?v=qYIhA3O9Nz0

Babai, Laszlo. 2015/2016. "Graph Isomorphism in Quasipolynomial Time". https://arxiv.org/abs/1512.03547

Cormen et al.. 2009. *Introduction to Algorithms.* MIT Press.

Knuth, Donald. 2018. "Donald Knuth at Lulea University of Technology". https://www.youtube.com/watch?v=74BfHoE66rc (source front illustration)

Lipschutz, Seymour and Lipson, Marc. 2009. *Discrete Mathematics.* McGraw-Hill.

Rosen, Kenneth. 2011. *Discrete Mathematics and Its Applications.* McGraw-Hill.

Notes

[1] Sweden's northmost big city.

[2] "*Professor emeritus Donald Knuth, Q&A-session at Lulea University of Technology on January 12, 2018.*" according to the description of its video recording. (See Knuth 2018.)

[3] Richard Feynman was famous among many other things for the saying attributed to him (of many variations) : "*What I cannot build, I cannot understand.*" This seems relevant later when Knuth says that one of his first reflexes when approaching a problem is to translate it into code of some sort or a program.

[4] Donald Knuth shares this trait, a background in physics, with other computer scientists like Dennis Richtie or Richard Stallman.

[5] Milwaukee is part of the so-called "Midwest" (i.e. somewhere between California and New York...), located approx. 100 km North of Chicago.

[6] "*Graphs G(V, E) and G(V*, E*) are said to be isomorphic if there exists a one-to-one correspondence $f : V \longmapsto V^*$ such that $\{u, v\}$ is an edge of G if and only if $\{f(u), f(v)\}$ is an edge of G*. Normally, we do not distinguish between isomorphic graphs (even though their diagrams may "look different").*" (Lipschutz and Lipson 2009, ch. 8 "Graph Theory")

[7] Elsewhere, we find the following additional, useful information : "*It is often difficult to determine whether two simple graphs are isomorphic. There are n! possible one-to-one correspondences between the vertex sets of two simple graphs with n vertices. Testing each such correspondence to see whether it preserves adjacency and nonadjacency is impractical if n is at all large.*" (Rosen 2011, ch. 10 "Graphs")

[8] Babai 2015/2016.